Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 83 (2016) 537 – 544

Procedia Computer Science

The 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016)

Comparative Analysis of Prominent Middleware Platforms in the Domain of Ambient Assisted Living (AAL) for an Older Adults with Dementia (OAwD) Scenario

Rajjeet Phull[a], Ramiro Liscano[b,*], Alex Mihailidis[a]

[a]*Institute of Biomaterials and Biomedical Engineering, University of Toronto, 500 University Ave, Toronto, CANADA*
[b]*Dept. of Electrical, Computer, and Software Engineering, 2000 Simcoe Street North, Oshawa, CANADA*

Abstract

Diversification of application areas, technologies, and computational techniques in the Ambient Assisted Living (AAL) domain alongside the need for a balance between reproducibility and customizability for the heterogeneous end-user groups has led towards the development of domain-specific middleware platforms, which are software frameworks that facilitate integration and communication amongst heterogeneous hardware and software components, so they can operate synergistically within a shared environment. Recent efforts that study such platforms have emphasized the need for improved evaluation methods, particularly feature-focused scenario-based evaluations. Thus, the objective of this work was to perform a comparative analysis of two prominent AAL middleware platforms and their programmability for Older Adults with Dementia (OAwD) to ultimately derive guidelines for the development of AAL systems. Mapping of a platform-independence use-case scenario to both platforms, hinted towards the need for graphical-user interfaces over text-based interfaces and application area-specific modules.

Keywords: Ambient Assisted Living; Middleware; Unified Modeling Language; Older Adults with Dementia

* Corresponding author. Tel: +1-905-721-3086
E-mail address: rliscano@ieee.org

doi:10.1016/j.procs.2016.04.252

1. Introduction

According to the universal reference model for Ambient Assisted Living (AAL)[1], AAL systems can be defined as socio-technical systems that consist of networked artefacts (i.e. sensors & actuators) embedded in an AAL space to provide various types of AAL services for the wellbeing of the assisted persons (e.g. the elderly or disabled). AAL services (or applications) are specific functions that use the input sensor data to make actions which facilitate assistance or boost social integration for the assisted person[1]. AAL space is the smart environment equipped with the networked artifact that allow for the provision of AAL services.

Advancements in the AAL field are occurring in a broad range of applications areas such as cognitive orthotics, emergency detection, continuous health and activity monitoring, therapy, and emotional well-being[2]. Applications based on static smart environments incorporate ambient or environment sensors such as cameras, radio frequency identification, microphones, passive infrared sensors, and pressure sensors while mobile-based applications (i.e. e-textiles, vital sign detection) use wearable sensors like accelerometers, gyroscopes, ECG, and pulse oximeters[2]. Sensory information is processed using several types of computational techniques such as context modelling, activity recognition, anomaly detection, planning, and location identification[2].

The diversity and heterogeneity of assisted persons (end users of AAL services) is one of the main challenges when developing AAL systems. Persons supported by AAL systems can vary from one another in terms of age, cognitive abilities, preferences, physical capabilities, perceptions on technological aid, and environmental conditions. Therefore, the AAL solutions must be customizable and adaptive to the needs of its end users. Also, many of the developed AAL systems target only a subset of the entire user population and adopt methods and tools that are not easily transferable to other projects, resulting in fragmentation within the AAL field[3]. To ensure technical and financial feasibility of large, complex AAL systems, AAL middleware platforms are being developed. Middleware can be described as the systems of systems that resides between the operating system and the application layer. Requirements for AAL middleware are on the one hand mechanisms for personalized user interfaces and on the other hand integration of different backend technologies.

Many AAL middleware platforms have been developed[4,6,7,11,12] and the current contribution to the literature is primarily in the area of assessing the suitability of these AAL platforms on realistic AAL scenarios. With this in mind the objective of this work was to perform a comparative analysis of two prominent AAL middleware platforms and their programmability for Older Adults with Dementia (OAwD). The contribution of this paper is a set of guidelines and/or recommendations for future AAL systems to help improve adoption rates of the platforms by AAL application developers, and render the platforms more useful to a larger selection of AAL project types. Even though only two AAL platforms were compared one of these platforms, UniversALL[4] is a consolidation of the features from several AAL systems and a conscious decision was made to not include those AAL systems that were incorporated into UniversAAL.

This paper is organized as follows. Section 2, is an overview of work related to AAL systems. Section 3, leverages the scenario-based approached in Software Engineering to a particular assisted living scenario of an elderly person struggling with dementia trying to make a cup of tea. Section 4, discusses the platform specific models and methodologies and how they affect the programming of the tea making scenario. Finally, section 5 presents guidelines to help AAL developers based on the experience of developing the tea making scenario.

2. Related Work

Over the past decade, multiple AAL-related platforms have been developed, many of which have been consolidated into recent projects and/or have been discontinued. Many legacy projects have been consolidated as input projects for UniversAAL[4]. Other main frameworks and open solution platforms are highlighted in a recent literature survey of AAL by Memon et al.[5] which includes OpenCare[6] and AmiVital[11]. Additionally, Memon et al.[5] incorporated an email-based survey to create a detailed list of the contemporary AAL systems and platforms. From the presented list of AAL platforms, the ones that were based on a middleware were AALuis[12], HOMER[7], and OpenCare[6]. Of these platforms, HOMER[7] and UniversAAL[4] are the only prominent AAL-specific middleware platforms that are available under open source license and are still in development.

The only evaluation of AAL platforms was carried out in 2011 by Antonino et al.[8] that was based on the architectural-based quality attributes of the platform's software architecture. The methodology for evaluating the included platforms involved interviewing platform developers to understand the features and functions their platforms provided in respect to the different quality attributes or non-functional properties. Results from this evaluation yielded that UniversAAL, at the time, was the platform that highly addressed all the measured quality attributes. Antonino et al. [8] also emphasized that scenario-based evaluations are required for future evaluations, and for contributions towards developing more effective AAL middleware platforms.

UniversAAL and HOMER were selected for this comparison because they were the only available AAL-specific middleware platforms under open source license, and are actively in development. Other platforms were omitted from the comparative analysis, primarily due to the lack of development-related activities, and developer support. Suggesting improvements on active projects takes precedence in order to make useful and immediate impact on the developer community.

3. Scenario-based Evaluation

The use of scenarios is a common technique to help understand the needs of an aging population. It is commonly associated with a persona that is defined as a fictitious representation of target users that is specific and concrete[13]. A scenario highlights a specific part of the persona's day or experience; a snap shot presented in the form of narratives. Scenarios are also commonly used in the design of software applications because they help to capture how a design is used, they focus on the end user, problems and needs may surface that were not originally considered, and allow for the exploration and comparison of multiple ideas.

3.1. Test-case Scenario

The specific scenario or use case selected, for evaluating the platforms, was an audio prompting system that aims to assist OAwD or assisted person in simple meal preparation. This use case is selected based on study findings, conducted on 106 caregivers of dementia patients, which emphasized that assistance in meal preparation is one the most demanding functionalities of future AAL systems[9]. Furthermore, this scenario also build upon a recently conducted study that investigated the effectiveness of a prompting assistive robot in helping dementia patients make a cup of tea[10]. The scenario has also been used to animate how an AAL system could be programmed by a family member and used in a home to help an older adult with dementia and posted on the Internet[14] to be shared with the AAL community.

The tea making scenario is rather simple consisting of two concurrent activities that prepare the utensils and boil the water followed by the actual preparation of the tea. This process is captured in the activity diagram shown in Figure 1. While these tasks are carried out, sensors within the environment detect conditions that imply the state of the environment and the completion of specific tasks. When the OAwD patient requires help in completing the tea making process it requests for support. This request triggers an audio prompt that directs the AAL application to help the OAwD complete a pending task. From this point on we refer to this application as the OAwD-AAL application.

3.2. The Tea making Activities

The activity diagram of the tea making process is shown in Figure 1. This process consists of 14 individual activities that have specific sequencing and concurrencies. The color scheme indicate sets of activities corresponding to concurrent major activities. The green, for example, are activities that represent filling the kettle with water and having it boil. Preparing the cup is shown by the yellow activities. Blue activities merge the outcome of the yellow and green activities, in which the prepared cup and the boiled water are combined to finish making a cup of tea. The initial split signifies that the green and yellow activities can be carried out in parallel, but both paths must be complete in order for the blue colored activities to be completed. This formal model of the activities required to prepare tea forms the foundation of the OAwD-AAL application and the first step in formally capturing the scenario. The design and resolution of the activity diagram is primarily driven by the following guidelines: They

tend to be singleton actions that are part of a larger activity and the start and/or end of the activity can be detected by a set of sensors embedded in the environment.

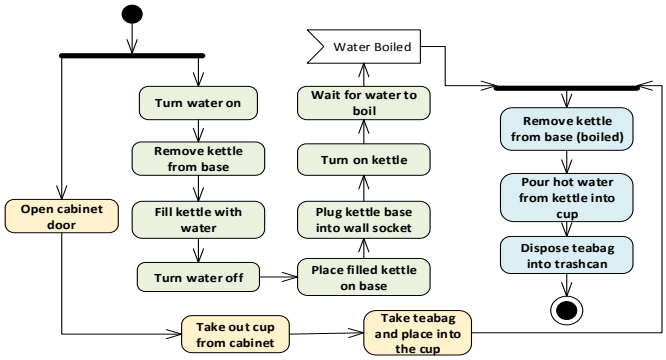

Figure 1. An activity diagram for the tea making process.

Since each activity can be detected by the sensors in the system it is possible to associate with each activity a state such that the OAwD-AAL application can determine the proper support prompt for the OAwD. Hence, this activity diagram only includes activities that can be monitored with embedded sensors.

3.3. Environmental State

In an AAL system, sensory data is captured from the AAL space for the applications within the system to understand its state and make appropriate adjustments. For the OAwd-AAL application the state captured is used to track the progress made by the OAwD in making a cup of tea, following the activity diagram. For the purpose of this simulated test-case, embedded sensors were theorized for the kettle, faucet, cabinet, teabag container, and cup. Figure illustrates the different binary state machines for individual objects that are required in the tea making process. Since the AAL middleware platforms cater to binary, simple sensors (high level data) and since the hardware connection layer is not considered in this evaluation, the included sensors have at most two states that oscillate between one another when a certain threshold has been reached. For instance, the state of the faucet is switched to ON when its embedded sensor reaches a specific threshold, indicating that the water is running. Together, the state of all these individual devices helped determine the progress made by OAwD within the tea making progress, and select the most appropriate audio prompt in case the OAwD requested assistance.

3.4. Audio Prompting System (States of the AOwD-AAL Application)

The main component of the OAwD-AAL application is the audio prompting system. As the state of the environmental sensors is being recorded the OAwD-AAL is progressing through the tea making process. Assistance can be demanded at any moment by pressing an assist button.

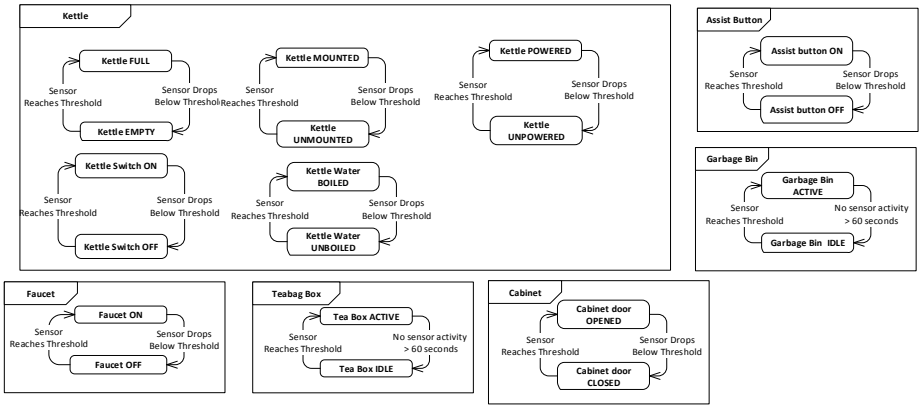

Figure 2. Individual state diagrams of the different sensor devices involved in the audio prompting system for tea making

Table 1 maps the 14 individual audio prompts (each prompt is associated with an activity) to the states of the OAwD-AAL application, hence this is a mapping from activity to states. The colour coding maps to the set of activities introduced in Figure . Blank cells represent the "don't care" condition, in which the state of the particular device is not checked. Because of concurrent nature of the tea making process it is not possible to have unique sensor values for concurrent activities. For example, if one looks at the activities for taking out a cup from the cabinet (yellow rows in Table 1) and the core activities for making tea (the green rows in Table 1) once can see that states of the cabinet and cup are not relevant for the tea making process. For this reason the OAwD-AAL application also uses knowledge of the prior state to determine the unique state of the OAwD. The assumption is that the OAwD was performing a particular activity and was moving on to the next activity and forgot what he/she was doing. This is not a perfect solution to this challenge and was not the intent of the research but it has surfaced as a "key" issue that cannot be accurately determined. Basically one has no way to read the mind of the OAwD.

Table 1. Selection of audio prompts based on the different sensor states. A blank cell indicates that the state of that particular sensor is not relevant.

Index	Audio Prompt	Faucet	Kettle					Cabinet	Cup		Teabox	Garbage bin
		ON/OFF	Full/ Empty	Plugged/ Unplugged	Mounted/ Unmounted	Boiled/ Unboiled	Powered/ Unpowered	Open/ Close	Outside/ Inside	Empty/ Full	Active/ Idle	Active/ Idle
1	Turn on the water	OFF	EMPTY			UNBOILED						
2	Remove Kettle from base		EMPTY		MOUNTED	UNBOILED						
3	Fill kettle with water	ON	EMPTY		UNMOUNTED	UNBOILED						
4	Turn the water off	ON	FULL			UNBOILED						
5	Place kettle on base		FULL		UNMOUNTED	UNBOILED						
6	Plug kettle into wall socket			UNPLUGGED		UNBOILED						
7	Turn the kettle on		FULL	PLUGGED	MOUNTED	UNBOILED	UNPOWERED					
8	Open the cabinet							CLOSE	INSIDE			
9	Take out cup from cabinet							OPEN	INSIDE			
10	Take out teabag and put it into cup								OUTSIDE		IDLE	
11	Remove kettle from base (boiled)		FULL		MOUNTED	BOILED	UNPOWERED					
12	Pour hot water from kettle into cup		FULL		UNMOUNTED	BOILED	UNPOWERED		OUTSIDE	EMPTY		
13	Put teabag into garbage bin		EMPTY			BOILED				FULL		IDLE
14	Wait for the water to boil		FULL			UNBOILED	POWERED					

3.5. The AOwD-AAL Application Behaviour

The behaviour of the audio prompting system is shown in Figure 3. The generic state machine of the audio prompting system for aiding an OAwD make a cup of tea. The core of this state machine are the following four state

types: *wait mode*, *active mode*, *audio prompt*, and *call caregiver*. The *wait mode* state is the default state of the system, as long the assist button remains in the *OFF* state. When the assist button transitions into the *ON* state, the application enters active mode. Upon entry into this state, the counter is incremented. The *counter* is used to determine the number of audio prompts the prompting system has delivered to the assisted person. When the *counter* reaches an arbitrary number (5 in our case), the state transition occurs from the *active mode* to *call caregiver* state that triggers an event to call the caregiver.

Once the audio prompt has been delivered to the OAwD, the application continuously (every second) queries the states of sensors that are relevant for the issued audio prompt. If the appropriate action is taken by the OAwD that continues the tea making process, the state machine transitions back to the *wait mode* and switches the audio button to the *OFF* state. On the contrary, if no action is detected within 10 seconds of issuing the audio prompt, the application enters *assist mode* and the *counter* is incremented. Depending on the *counter* value, the caregiver call is invoked or the application goes through the sequential checking of sensor states to determine the appropriate audio prompt. The ordered checking of sensor states for each audio prompt is based on its index.

4. Platform-specific Implementations

The platform-independent model was implemented on both, UniversAAL and HOMER, to understand ways the different platform's features, functions and constraints influenced the application design. For each platform there are particular design paradigms that force the application developer to convert the platform independent models presented in Section 3. We outline these points.

4.1. UniversAAL

UniversAAL's runtime environment is based on a bus architecture, specifically the context, service, and user-interface buses, that exchange messages between the platform instances using the publish-subscribe mechanism. To use these buses, a platform-specific ontology was imposed on each platform instance though the bus wrappers. A UML tool was provided to create domain specific ontologies, however, actual usage of the tool demanded an in-depth knowledge of the ontology tree and coding patterns for each type of bus wrapper. For our use-case, this UML tool integrated the application-specific sensors for the cup, kettle, cabinets, and teabag container into data models usable by any other platform instance.

The tea making use-case scenario was divided into different components, or object instances, that can be implemented in the platform. Sensors and actuators, hypothetically embedded into the surrounding objects, were categorized as context publishers and service callees, respectively. The audio prompting aspect of the use-case encapsulated a set of context subscribers for the different sensors that published their status on the context bus, service callers that elicited specific functionality from actuators via the service bus, and UI callers for delivering the audio prompt via the UI bus to an appropriate UI handler.

Besides the UML tool for creating custom domain-specific ontology and basic code templates for each bus wrapper, the UniversAAL platform, being a text-based interface, required the entire application (including its behavior logic as introduced in Figure 3) to be coded in Java. The majority of the development effort was spent on integrating the platform-specific and custom ontology to bus wrappers of each platform instance, rather than the application logic itself. The cost-benefit of using ontologies seemed questionable at the scale of the implemented use-case, but can save effort and ease communication when multiple use-cases require interaction.

4.2. HOMER

The HOMER platform is composed of two main GUI components which are: (1) the flat designer that allows for the creation of AAL space along with the placement and configuration of sensors and actuators, and (2) the scenario editor that consists of multiple, concurrently running finite state machines. State transitions and actions occur in response to events generated by the flat designer or other state machines.

Implementing the use-case on the HOMER platform highlighted some key constraints. Firstly, the flat designer supported only stationary devices and not mobile-based ones as it closely followed the ISO-11703[15] home

automation standard. Having mobile devices being represented as static ones is possible, however, the GUI wouldn't be an accurate representation of actual AAL space. Second, multiple sensor or actuator types were unable to be combined and expressed as single entities or devices or even occupy the same region on the flat designer. For the scenario editor, status of sensors and actuators didn't persist as state transitions were only driven by events. The only method to persist past sensor/actuator events was to create their own individual finite state machine and change the system state variable corresponding to that device. Thus, the value of the system state variable would be used as conditions for state transitions in other state machines. For instance, a contact closure sensor can manipulate a system variable named "Cabinet A" as either open or close depending on the sensor event detected. Other state machines can then use "Cabinet A" variable to make decisions, such as picking the appropriate audio prompt.

Another major constraint was that only a single condition was allowed per state transition, where the audio prompts required multiple condition checks for multiple sensors when transitioning from active mode state to an audio prompt state. To compensate, a separate state machines was required to continuously monitor the devices for events and change a system state variable that dictated the progression within the process of making tea by the OAwD-AAL application. This state machine was used by another state machine that was responsible for evoking audio prompts in response to the assist request. Selecting correct prompts was determined by the state of the system that is known by the sensors of the system as well as the generic state machine shown in Figure 3.

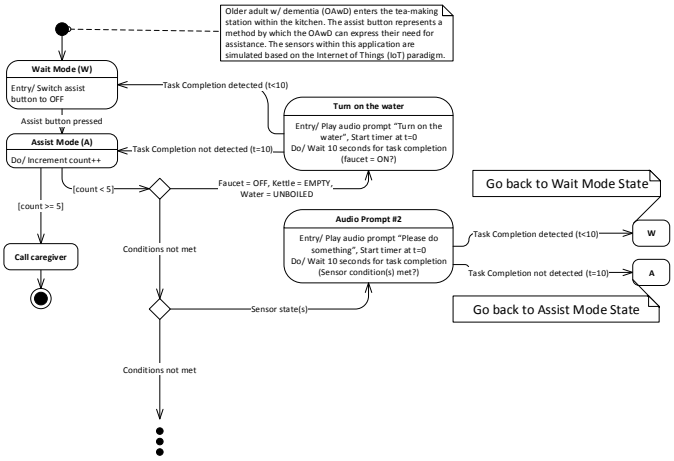

Figure 3. The generic state machine of the audio prompting system for aiding an OAwD make a cup of tea.

Unlike UniversAAL, HOMER provided direct features for implementing the business logic of the application, however, multiple constraints made it difficult to fully realize the use-case scenario as designed.

5. Observations and Guidelines for AAL Developers

Determining the "ideal" application developer of these AAL middleware platforms is not so clear. UniversAAL's text-based interface points to the software developer as the primary creator of the AAL applications as they are responsible for its code creation. Moreover, the "AAL deployers" is recognized as the secondary users as they are the stakeholders that install the AAL application in the AAL spaces. For HOMER, their end-user is not explicitly stated, however, it is implied that the deployers of the AAL application interact with the GUI and the software developers make modifications to the platform itself. As of now, creation of AAL application would require close interaction between the software/hardware developers and an expert within the AAL-domain or the application developer. Based on these facts, the following key directions of investigation are suggested.

That the current AAL middleware platforms should aim for higher levels of abstraction for the creation of its AAL applications through the use of graphical user-interfaces as opposed to text-based interfaces (i.e. text-based IDEs, command-line interfaces etc.) GUI-based designs can potential transition the role of application developer from the traditional hardware/software developer towards the medical staff, domain-experts, and deployers. This can be achieved by deriving add-ons, as mentioned previously, that rely on a GUI for the provision of its functionality. The benefits include expanding the AAL middleware platform user base to individuals who work more closely with the ultimate end-users of AAL applications, the elderly and disabled. Additionally, these individuals can potential reduce the involvement of core hardware/software developers, thereby reducing cost and effort, while creating applications that better match the business goals of this domain.

That a common modeling language be developed to bridge the gap between home-care personnel and software development. It is clear that AAL systems will be programmed or customized by home-care personnel and not software developers. As per demonstrated in this work, this requires a simple but effective way to model a process and the association of the states of the sensors to the activities of a task.

Lastly, the requirement for customizability and repeatability for the application creation is higher than other domain areas as the end-users are diverse in many aspects. From the perspective of repeatability, the goal should be increase the abstraction level of the platform tools to a point where it maintains a balance between standardization of AAL solutions that cater to the masses and allow individuals with less technical expertise to create these solutions. From the perspective of customizability, the level of alteration allowable from the platform should only be enough to cover the diversity of the end-users.

Acknowledgements

This work was possible thanks to the financial support of the AAL-WELL: Ambient Assistive Living Technologies for Wellness, Engagement, and Long Life project.

References

1. W. P. Tazari M.-R,Furfari F,Fides Valero Á,Hanke S,Höftberger O,Kehagias D,Mosmondor M,Wichert R, “The universal Reference Model for AAL,” Handb. Ambient Assist. Living, vol. 1, pp. 610 – 625, 2012.
2. P. Rashidi and A. Mihailidis, “A Survey on Ambient-Assisted Living Tools for Older Adults,” IEEE J. Biomed. Heal. Informatics, vol. 17, no. 3, pp. 579–590, May 2013.
3. F. Palumbo, P. Barsocchi, F. Furfari, and E. Ferro, “AAL Middleware Infrastructure for Green Bed Activity Monitoring,” J. Sensors, vol. 2013, pp. 1–15, 2013.
4. S. Hanke, C. Mayer, O. Hoeftberger, H. Boos, R. Wichert, M.-R. Tazari, P. Wolf, and F. Furfari, “universAAL - An Open and Consolidated AAL Platform,” Ambient Assited Living 4 Dtsch. AALKongress, pp. 127–140, 2011.
5. M. Memon, S. R. Wagner, C. F. Pedersen, F. H. A. Beevi, and F. O. Hansen, “Ambient assisted living healthcare frameworks, platforms, standards, and quality attributes.,” Sensors (Basel)., vol. 14, no. 3, pp. 4312–41, Jan. 2014.
6. S. Wagner and C. Nielsen, “OpenCare project: An open, flexible and easily extendible infrastructure for pervasive healthcare assisted living solutions,” 2009 3rd Int. Conf. Pervasive Comput. Technol. Healthc., 2009.
7. T. Fuxreiter, C. Mayer, S. Hanke, M. Gira, M. Sili, and J. Kropf, “A modular platform for event recognition in smart homes,” 12th IEEE Int. Conf. e-Health Networking, Appl. Serv. Heal. 2010, 2010.
8. P. O. Antonino, D. Schneider, C. Hofmann, E. Y. Nakagawa, and F. Iese, “Evaluation of AAL Platforms According to Architecture-Based Quality Attributes,” pp. 264–274.
9. S. Czarnuch and A. Mihailidis, “The design of intelligent in-home assistive technologies: Assessing the needs of older adults with dementia and their caregivers,” Gerontechnology, vol. 10, no. 3, 2011.
10. M. Begum, R. Wang, R. Huq, and A. Mihailidis, “Performance of daily activities by older adults with dementia: The role of an assistive robot,” IEEE Int. Conf. Rehabil. Robot., 2013.
11. P. Abril-Jiménez, “Design Framework for Ambient Assisted Living Platforms,” Univers. Access …, pp. 139–142, 2009.
12. C. Mayer, M. Morandell, S. Hanke, J. Bobeth, T. Bosch, S. Fagel, M. Groot, K. Hackbarth, W. Marschitz, C. Schüler, and K. Tuinenbreijer, “Ambient Assisted Living User Interface”, Everyday Technology for Independence and Care, AAATE 2011, pp. 456 – 463.
13. J. Pruitt and T. Adlin, The essential persona lifecycle: Your guide to building and using personas, Burlington, MA: Morgan Kaufmann. 2010.
14. J. Hoey and A. Hwang, DIYSmartHome, https://www.youtube.com/watch?v=zKpRXrhR9Ig, accessed Jan 30, 2016.
15. ISO/IEEE 11073-20601:2010, Health informatics - Personal health device communication - Part 20601: Application profile - Optimized exchange protocol, 2010.